WebFormulate: A Web-Based Visual Continual Query System

Jennifer Leopold, Meg Heimovics, and Tyler Palmer

University of Kansas
Department of Electrical Engineering and Computer Science
Welcome

T-Shirts, Caps, and Big Foam-Fingers will be available in the lobby after the show.
Presentation Overview

- What seems to be the problem?
- What do we need?
- What do you mean by a “visual” system?
- What is a “continual” query?
- How does WebFormulate work?
- How is WebFormulate going to change the world?
- What’s next?
Introduction

• *WebFormulate*: Web-based *visual continual query system*

• Addresses problems associated with formulating temporal ad hoc analyses over networks of heterogeneous, frequently-updated data sources
What distinguishes *WebFormulate*?

• provides necessary facilities to perform *continual* queries

• develops and maintains dynamic links such that Web-based computations and reports automatically maintain themselves

• specifically designed for users of spreadsheet-level ability, rather than professional programmers
What seems to be the problem?

Considerable amounts of Internet-distributed data go unnoticed and unutilized, particularly frequently-updated, Internet-distributed databases.
Infrastructural constraints:

(1) Internet-distributed computing world was developed for one-time distribution of information, not the continuous flow of process communication
Infrastructural constraints:

(2) Heterogeneous nature of database systems makes it difficult to have a single user interface that handles different:

- connectivity protocols,
- schema metadata, and
- SQL syntax
Infrastructural constraints:

(3) Standards infrastructure for data exchange is underdeveloped
Infrastructural constraints:

(4) “Public programmers” lack necessary skills to access, query, and analyze data from heterogeneous, Internet-distributed databases
What do we need?

“Public programmers” need one tool to:

• simultaneously query multiple, disparate databases

• track changes in those databases through time,

• automatically update user-specified computations and visualizations
What do you mean by a “visual” system?

• **Visual Programming Languages** (VPLs) use **more than one dimension** to convey semantics

• **Visual expressions** include diagrams, sketches, icons, demonstrations of actions performed by graphical objects, etc.

• Not just an editing shortcut to generate (textual) code!
The goals of a VPL:

(1) to make programming more accessible to some particular audience

(2) to improve the correctness with which people perform programming tasks, and/or

(3) to improve the speed with which people perform programming tasks

Elimination of text is not a goal of VPLs!
What is a “continual” query?

A query that monitors updates of interest and notifies the user of changes whenever an update reaches specified thresholds or some time limit is reached.

Example: Notify me weekly OR whenever the number of *Rana blairi* and *Rana catesbeiana* frogs increases by 10% since the last time the database was queried.
What is a “continual” query?

Expressed in terms of:

- SQL-like query
- trigger condition
- stop condition
- notification condition

First introduced by Terry et al. in 1992 for append-only databases
How does *WebFormulate* work?

- **Form-based** VPL with a Web-browser GUI
- Place cells on a form
- Define formulas for cell attributes using pointing, typing, and gesturing
- Attributes of other cells can be referenced
- Programs developed using “live” data
Example

Biologist in California

Database of frogs in Kansas

Weekly updates

Graduate students doing field work
Example

(1) Create a ‘database’ object and specify the URL of an Internet-accessible database

Database schema displayed as hierarchical tree of table and field names
Example

(2) Create a ‘database query’ object and construct a continual query equation

(i) List of fields to return in query results

(CQSELECT ( 
  FullTaxonName
  Latitude
  Longitude
) …)
(2) Construct a continual query equation

(ii) Conditional expression for selection criteria

(CQSELECT …
(AND
  (IN FullTaxonName (“Rana blairi” ”Rana catesbeiana”))
  (>= Latitude 37.8) …
  (>= Longitude 94.8)) … )
Example

(2) Construct a continual query equation

(iii) Order to sort query results

(CQSELECT ...
((ASC FullTaxonName )
...
)
Example

(2) Construct a continual query equation

(iv) Notification condition

(CQSELECT ...
(OR (WEEKLY)
(> (COUNT
(OR (FullTaxonName))
(+ (PREVCOUNT
(1000))))) ...)
(OR (FullTaxonName))
...)

FullTaxonName
Example

(2) Construct a continual query equation

(v) Trigger condition (i.e., how often database will be queried)

(CQSELECT … (DAILY) …)
Example

(2) Construct a continual query equation

(vi) Stop condition (i.e., termination of this continual query process)

(CQSELECT … (DATE 10 31 2002))
Example

(3) Submit continual query equation for evaluation

Results displayed as a table

<table>
<thead>
<tr>
<th>FullTaxonName</th>
<th>Latitude</th>
<th>Longitude</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rana blairi</td>
<td>38.2023</td>
<td>95.102</td>
</tr>
<tr>
<td>Rana blairi</td>
<td>38.203</td>
<td>95.145</td>
</tr>
<tr>
<td>Rana blairi</td>
<td>38.402</td>
<td>95.614</td>
</tr>
<tr>
<td>Rana blairi</td>
<td>38.403</td>
<td>95.145</td>
</tr>
<tr>
<td>Rana blairi</td>
<td>38.403</td>
<td>95.145</td>
</tr>
<tr>
<td>Rana blairi</td>
<td>38.502</td>
<td>95.0008</td>
</tr>
<tr>
<td>Rana blairi</td>
<td>38.502</td>
<td>95.203</td>
</tr>
<tr>
<td>Rana blairi</td>
<td>38.673</td>
<td>95.056</td>
</tr>
<tr>
<td>Rana blairi</td>
<td>38.673</td>
<td>95.056</td>
</tr>
</tbody>
</table>
Example

(4) Create a ‘graph’ object to plot occurrences by latitude and longitude

(Graph
  ([38.2...95.8]
   (black-circle))
  ([37.9...95.6]
   (black-square))))

Ranges of latitude and longitude values selected from ‘database query’ (table) object
Other Types of Graph Functions
Automatic Updating

Each time continual query processor returns updated results, affected cells will recalculate and redisplay.

Example: ‘Database Query’ object and associated “Graph” object display.
Automatic Updating

- If form **closed**, the next time it is opened, **most recent data** will be displayed
- Others can open a “**copy**” of the form
- “**Copies**” **updated** with recomputed data in same manner and timeframe as “**original**” form
- **Facilitates sharing** “live” form with others
System Architecture

Client submits continual query

Evaluation engine translates equation into SQL query

Query results returned as XML document

Query results returned as a table
Continual Query Processor

• Uses Enterprise Java Beans (EJBs) and EJB server technologies

• Automatically manages transactions, object distribution, concurrency, security, persistence, and resource management

• Portable and scalable framework
Installing a Continual Query

1. Client requests CQManagementBean
2. Server returns remote reference to CQManagementBean
3. Client requests query installation
4. Information persisted to backend database

Application Server
- CQManagementBean
- CQBean
- ScheduleBean

Database Server
- CQBean table
- ResultBean table
- ScheduleBean table
Processing a Continual Query

1. Daemon runs periodically to process scheduled queries.
2. Thread created to run a scheduled query or notify client of results.
3. ScheduleManagerBean locates queries that need to be run during this cycle.
4. CQBean retrieves query specifications.
5. Updated results are maintained in a ResultBean.

Diagram:
- Application Server
  - QueryBean
  - CQBean
  - ScheduleManagerBean
  - ResultBean
  - ScheduleBean
- Database Server
  - CQBean table
  - ResultBean table
  - ScheduleBean table
- Thread
  - Schedule Daemon (1)
How is *WebFormulate* going to change the world?

A Web-based data access + analysis tool that:

- Is **usable** by a diverse population
- Is **not restricted** to any one problem domain
- Will **improve** research productivity
How is *WebFormulate* going to change the world?

Motivate the development of Web-based continual query systems that apply J2EE methodology to achieve:

- scalability,
- platform-independence, and
- implementation-independence
What’s next?

End-user programming side:

• Formal usability studies to evaluate usability and usefulness

• Inclusion of other forms of input (e.g., voice browsers)
What’s next?

Continual query processing side:

• **Performance evaluation** of the continual query processor

• **Extension to other types** of Web-accessible data
Summary

It is important not only that information be accessible to the public, but that the same public be able to combine this information into effective analyses.
Summary

Academic and commercial use of Internet-distributed databases has been hindered by:

• **Incomplete, dynamic, and unknowable structure of databases**
• **Lack of standards** for data model and query representations
• **Inability of non-programmers** to perform continual queries with automatic update of computations
Summary

WebFormulate addresses those problems allowing “public programmers” to:

• Query multiple, disparate databases
• Track changes through time
• Automatically update computations & visualizations
This work has been supported by NSF under award DBI-9905760

(Division of Biological Infrastructure)
The End!